08

Ignorando archivos

Transcripción

[00:00] Hola a todos. Bienvenidos de nuevo a este curso de Git. En el último vídeo creamos un archivo vacío, solo para ejemplificar que podría ser un archivo que no queremos monitorear, un archivo de configs de IDE. ¿Cómo puedo hacer para que Git ignore ese archivo?

[00:18] Vamos a nuestra consola y lo vamos a ejecutar git status, damos enter y vemos que ahí tenemos en este caso una carpeta que nosotros no queremos que Git monitoree. Entonces, ¿cómo puedo decir: "Git, está bien que esa carpeta esté ahí pero yo no la quiero monitorear realmente"?

[00:38] Bueno, lo que podemos hacer es crear un archivo especial de Git llamado Git ignore. Entonces, vamos a crear manualmente acá dentro de la carpeta de nuestro repositorio, un archivo llamado punto, es importante ese punto, .gitignore. Damos enter, y acá es donde nosotros colocamos cuáles archivos y carpetas no queremos que se tomen en cuenta dentro de Git.

[01:11] En este caso, nosotros tenemos una carpeta, entonces lo que vamos a hacer es colocar carpeta/, esa barra acá identifica que son carpetas. Y si yo tuviera un archivo que no quiero colocar, por ejemplo este a.config si yo lo coloco dentro de Bruno por ejemplo, yo no quiero que gitignore tome ese a.config, entonces puedo colocar a.config, doy enter y ahí con eso damos, guardamos, salvamos.

[01:55] Ahora si yo voy a nuestro Git Bash y hago un Git status, doy enter, vemos que ya no existe, ya no está tomando esa carpeta que está en rojo. No está

tomando el archivo a.config, pero vean que sí se agregó un .gitignore. Este archivo .gitignore sí es necesario que nosotros lo agreguemos a nuestro repositorio.

[02:12] Entonces lo que vamos a hacer es agregar ese archivo con git add, espacio, .gitignore. Acá podemos comenzar con .git, apretar la tecla tab de nuestro teclado una sola vez y se va a autocompletar el nombre del archivo. Damos enter. Vamos a hacer un git commit, espacio, -m, espacio, abro comillas y colocamos: "Agregando gitignore", cerramos comillas, damos enter y ahora ya se agregó.

[02:48] Si hacemos un Git status, solo para ver lo que está ocurriendo, nos dice que no hay nada para agregar. Si ahora hacemos un git log, espacio, --oneline, enter, vemos que lo último que hemos hecho es agregar el gitignore. Bueno, ahora puede ser que ustedes se estén preguntando: ¿En qué momento genero un commit? ¿Cuándo debería generarlo? ¿Solo al final del proyecto cuando termino todo o cada una línea de código tengo que generar un commit?

[03:19] El momento realmente es un asunto bastante extenso y que genera discusiones. En reglas generales, hay un consenso que dice que nunca hay que hacer un commit de un código que no funciona. Por ejemplo, vamos a nuestro archivo y por ejemplo supongamos que yo estoy comenzando ahora en una nueva línea, coloco li, cierro ahí.

[03:48] Estamos modificando acá el archivo y vamos a colocar un nombre de un curso nuevo, por ejemplo Nuevo curso, aún no termino de cerrar la línea y esto no funciona. Y por ejemplo tengo una reunión en este momento o mi jefe me llama y me dice: "Mirá, tengo una cosa urgente que necesito que hagas". ¿Qué hago? ¿Hago un commit? Bueno, en realidad no, porque como hemos dicho nunca se tiene que hacer un commit de un código que no está funcionando.

[04:15] Tu código tiene que funcionar siempre que se agrega un commit. Entonces, ¿voy a generar un commit al final del proyecto? No. En cada

alteración significativa se tiene que generar un commit. Hay personas que dicen que al final del día hay que hacer un commit, otras dicen que cada pequeña alteración se tiene que hacer un commit, pero en realidad no hay una regla sobre cuándo tenemos que commitear pero existen recomendaciones.

[04:43] Siempre que una pequeña alteración se agregue, hay que commitear. Cuando se corrige un bug, hay que hacer un commit, para que al final del todo, un conjunto de commits generen nuestro sistema como un todo y no un único commit. Solo quería dejar esa recomendación sobre cuándo commitear y cuándo no.

[05:06] Entonces como regla general, por favor nunca commiteen nada en un código que no funciona. Entonces voy a dar un clear aquí y a lo largo de esta aula vimos cómo crear un repositorio con git init, vimos también cómo transformar nuestra carpeta en un repositorio, además vimos cómo visualizar el status de nuestro repositorio con git status, vimos también cómo agregar archivos con git add y luego guardarlos con git commit -m y luego nuestro mensaje.

[05:51] Además de eso vimos cuál fue el histórico de nuestros commits con git log. Cómo dejar de monitorear determinado archivo utilizando un archivo llamado .gitignore. Entonces ya vimos varias cosas y ya podemos comenzar a trabajar con cosas interesantes con control de versiones. Pero ahora, ¿cómo comienzo a trabajar en equipo? ¿Cómo comparto mi trabajo? ¿Cómo envío ese archivo a otras personas del equipo?

[06:26] Vamos a conversar sobre eso en nuestra próxima aula. Entonces hagan los ejercicios, pregunten cualquier duda que tengan y luego de eso, vamos al próximo capítulo.